Positive first-order logic on words and graphs

Denis Kuperberg

CNRS, LIP, ENS Lyon, Plume Team

Birmingham Theory Seminar
24 November 2023

First-Order Logic (FO)

Signature: Predicate symbols $(P_1, \ldots, P_n)$ with arities $k_1, \ldots, k_n$.

Syntax of FO:

$$\varphi, \psi := P_i(x_1, \ldots, x_{k_i}) \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \neg\varphi \mid \exists x.\varphi \mid \forall x.\varphi$$

First-Order Logic (FO)

Signature: Predicate symbols $(P_1, \dots, P_n)$ with arities $k_1, \dots, k_n$.

Syntax of FO:

$$\varphi, \psi := P_i(x_1, \dots, x_{k_i}) \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \neg\varphi \mid \exists x.\varphi \mid \forall x.\varphi$$

Semantics of φ:
Structure $(X, R_1, \dots, R_n)$ is accepted or rejected.

First-Order Logic (FO)

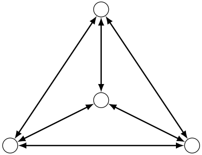
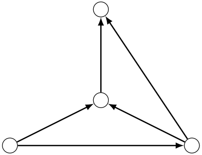
Signature: Predicate symbols $(P_1, \dots, P_n)$ with arities $k_1, \dots, k_n$.

Syntax of FO:

$$\varphi, \psi := P_i(x_1, \dots, x_{k_i}) \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \neg\varphi \mid \exists x.\varphi \mid \forall x.\varphi$$

Semantics of φ:
Structure $(X, R_1, \dots, R_n)$ is accepted or rejected.

Example: Directed graphs: one binary predicate E.

Graph class	Cliques	No node points to everyone
Formula	$\varphi = \forall x.\forall y.E(x,y)$	$\psi = \neg\exists x.\forall y.E(x,y)$
Example graph	 Model of φ	 Model of ψ

Positive versus Monotone

Goal: Understand the role of negation in FO.

Positive versus Monotone

Goal: Understand the role of negation in FO.

Positive formula: no $\neg$

Positive versus Monotone

Goal: Understand the role of negation in FO.

Positive formula: no $\neg$

Monotone class of structures: closed under adding tuples to relations.

For graph classes: monotone $=$ closed under adding edges.

Example: graphs containing a triangle.

Positive versus Monotone

Goal: Understand the role of negation in FO.

Positive formula: no $\neg$

Monotone class of structures: closed under adding tuples to relations.

For graph classes: monotone $=$ closed under adding edges.

Example: graphs containing a triangle.

Monotone formula: defines a monotone class of structures.

Positive versus Monotone

Goal: Understand the role of negation in FO.

Positive formula: no $\neg$

Monotone class of structures: closed under adding tuples to relations.

For graph classes: monotone = closed under adding edges.

Example: graphs containing a triangle.

Monotone formula: defines a monotone class of structures.

Fact: φ positive $\Rightarrow$ φ monotone.

Positive versus Monotone

Goal: Understand the role of negation in FO.

Positive formula: no $\neg$

Monotone class of structures: closed under adding tuples to relations.

For graph classes: monotone = closed under adding edges.

Example: graphs containing a triangle.

Monotone formula: defines a monotone class of structures.

Fact: φ positive $\Rightarrow$ φ monotone.

What about the converse ?

Positive versus Monotone

Goal: Understand the role of negation in FO.

Positive formula: no $\neg$

Monotone class of structures: closed under adding tuples to relations.

For graph classes: monotone = closed under adding edges.

Example: graphs containing a triangle.

Monotone formula: defines a monotone class of structures.

Fact: φ positive $\Rightarrow$ φ monotone.

What about the converse ?

Motivation: Logics with fixed points.
Fixed points need monotone φ.
$\rightarrow$ positive φ, syntactic condition.

Lyndon's theorem

Theorem (Lyndon 1959)

If φ is monotone then φ is equivalent to a positive formula.

On graph classes: FO-definable+monotone $\Rightarrow$ FO-definable without $\neg$.

Lyndon's theorem

Theorem (Lyndon 1959)

If φ is monotone then φ is equivalent to a positive formula.

On graph classes: FO-definable+monotone $\Rightarrow$ FO-definable without $\neg$.

⚠ valid with **infinite** structures.

Lyndon's theorem

Theorem (Lyndon 1959)

If φ is monotone then φ is equivalent to a positive formula.

On graph classes: FO-definable+monotone $\Rightarrow$ FO-definable without $\neg$.

valid with **infinite** structures.

What about **finite** structures ?

This was open for 28 years...

Lyndon's theorem

Theorem (Lyndon 1959)

If φ is monotone then φ is equivalent to a positive formula.

On graph classes: FO-definable+monotone $\Rightarrow$ FO-definable without $\neg$.

⚠ valid with **infinite** structures.

What about **finite** structures ?

This was open for 28 years. . .

Theorem: Lyndon's theorem fails on **finite** structures:

- [Ajtai, Gurevich 1987]
  lattices, probas, number theory, complexity, topology, very hard
- [Stolboushkin 1995]
  EF games on grid-like structures, involved

Lyndon's theorem

Theorem (Lyndon 1959)

If φ is monotone then φ is equivalent to a positive formula.

On graph classes: FO-definable+monotone $\Rightarrow$ FO-definable without $\neg$.

valid with **infinite** structures.

What about **finite** structures ?

This was open for 28 years...

Theorem: Lyndon's theorem fails on **finite** structures:

- [Ajtai, Gurevich 1987]
  lattices, probas, number theory, complexity, topology, very hard
- [Stolboushkin 1995]
  EF games on grid-like structures, involved
- [This work]
  EF games on words, elementary

Our results

Finite Model Theory:

Lyndon's theorem fails on

- Finite words
- Finite graphs
- Finite structures (elementary proof), several versions:
  - one monotone predicate
  - some monotone predicates
  - all monotone predicates = closure under surjective morphisms.

Our results

Finite Model Theory:

Lyndon's theorem fails on

- **Finite words**
- Finite graphs
- Finite structures (elementary proof), several versions:
  - one monotone predicate
  - some monotone predicates
  - all monotone predicates = closure under surjective morphisms.

Regular Language Theory:

Monotone FO languages	$\neq$	**Positive FO languages**
Algebraic characterization		Logical characterization
Decidable membership		Undecidable membership

FO on words, the usual way

Words on alphabet $A = \{a, b[, \dots]\}$: signature $(\leq, a, b[, \dots])$

$$\overset{a}{\bullet} \longrightarrow \overset{b}{\bullet} \longrightarrow \overset{a}{\bullet} \longrightarrow \overset{a}{\bullet} \longrightarrow \overset{b}{\bullet}$$

- $x \leq y$: position x before position y.
- $a(x)$: position x labelled by letter a

FO on words, the usual way

Words on alphabet $A = \{a, b[, \dots]\}$: signature $(\leq, a, b[, \dots])$

$$\overset{a}{\bullet} \longrightarrow \overset{b}{\bullet} \longrightarrow \overset{a}{\bullet} \longrightarrow \overset{a}{\bullet} \longrightarrow \overset{b}{\bullet}$$

- $x \leq y$: position x before position y.
- $a(x)$: position x labelled by letter a

Examples of formulas:

- $\exists x.a(x)$: Language A^*aA^*.
- $\exists x, y.(x \leq y \wedge a(x) \wedge b(y))$. Language $A^*aA^*bA^*$.

FO on words, the usual way

Words on alphabet $A = \{a, b[, \dots]\}$: signature $(\leq, a, b[, \dots])$

$$\overset{a}{\bullet} \longrightarrow \overset{b}{\bullet} \longrightarrow \overset{a}{\bullet} \longrightarrow \overset{a}{\bullet} \longrightarrow \overset{b}{\bullet}$$

- $x \leq y$: position x before position y.
- $a(x)$: position x labelled by letter a

Examples of formulas:

- $\exists x.a(x)$: Language A^*aA^*.
- $\exists x, y.(x \leq y \wedge a(x) \wedge b(y))$. Language $A^*aA^*bA^*$.

Theorem

First-order languages form a strict subclass of regular languages.

Example: $(aa)^*$ is not FO-definable. (Proof later)

Background: FO-definable languages

FO-definable languages are well-understood.

Background: FO-definable languages

FO-definable languages are well-understood.

Theorem (Schützenberger, McNaughton, Papert)

A language $L \subseteq A^*$ is FO-definable iff it is definable by:
Star-free expression $\Leftrightarrow$ LTL $\Leftrightarrow$ counter-free automaton $\Leftrightarrow$ ...

Background: FO-definable languages

FO-definable languages are well-understood.

Theorem (Schützenberger, McNaughton, Papert)

A language $L \subseteq A^*$ is FO-definable iff it is definable by:
Star-free expression $\Leftrightarrow$ LTL $\Leftrightarrow$ counter-free automaton $\Leftrightarrow \dots$

Intuition: FO languages are "Aperiodic": cannot count modulo
L aperiodic: There is $n \in \mathbb{N}$ such that $\forall u, v, w \in A^*$:

$$uv^n w \in L \Leftrightarrow uv^{n+1} w \in L.$$

Background: FO-definable languages

FO-definable languages are well-understood.

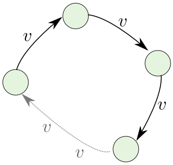
Theorem (Schützenberger, McNaughton, Papert)

A language $L \subseteq A^*$ is FO-definable iff it is definable by:
Star-free expression $\Leftrightarrow$ LTL $\Leftrightarrow$ counter-free automaton $\Leftrightarrow \ldots$

Intuition: FO languages are "Aperiodic": cannot count modulo
L aperiodic: There is $n \in \mathbb{N}$ such that $\forall u, v, w \in A^*$:

$$uv^n w \in L \Leftrightarrow uv^{n+1} w \in L.$$

$\Leftrightarrow$ Counter-free automaton: No cycle of the form:



Background: FO-definable languages

FO-definable languages are well-understood.

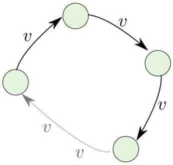
Theorem (Schützenberger, McNaughton, Papert)

A language $L \subseteq A^*$ is FO-definable iff it is definable by:
Star-free expression $\Leftrightarrow$ LTL $\Leftrightarrow$ counter-free automaton $\Leftrightarrow$ ...

Intuition: FO languages are "Aperiodic": cannot count modulo
L aperiodic: There is $n \in \mathbb{N}$ such that $\forall u, v, w \in A^*$:

$$uv^n w \in L \Leftrightarrow uv^{n+1} w \in L.$$

$\Leftrightarrow$ Counter-free automaton: No cycle of the form:



Corollary: FO-definability is decidable for regular languages.

FO on words, the “unconstrained” way

For now, a word is a structure $(X, \leq, a, b, \dots)$ where

- $\leq$ is a total order
- $a, b, \dots$ form a partition of X.

FO on words, the "unconstrained" way

For now, a word is a structure $(X, \leq, a, b, \ldots)$ where

- $\leq$ is a total order
- ~~$a, b, \ldots$ form a partition of X.~~

$a, b, \ldots$ now independent.

FO on words, the "unconstrained" way

For now, a word is a structure $(X, \leq, a, b, \dots)$ where

- $\leq$ is a total order
- ~~$a, b, \dots$ form a partition of X.~~

$a, b, \dots$ now independent.
$\rightarrow$ Words on alphabet $\mathcal{P}(\{a, b, \dots\})$:

$$\overset{\emptyset}{\bullet} \longrightarrow \overset{\{b\}}{\bullet} \longrightarrow \overset{\{a,b\}}{\bullet} \longrightarrow \overset{\emptyset}{\bullet} \longrightarrow \overset{\{b\}}{\bullet}$$

We will note $\Sigma = \{a, b, \dots\}$, and $A = \mathcal{P}(\Sigma)$ the alphabet.

- Useful e.g. in verification (LTL,...):
  independent signals can be true or false simultaneously.

FO on words, the "unconstrained" way

For now, a word is a structure $(X, \leq, a, b, \ldots)$ where

- $\leq$ is a total order
- ~~$a, b, \ldots$ form a partition of X.~~

$a, b, \ldots$ now independent.
$\rightarrow$ Words on alphabet $\mathcal{P}(\{a, b, \ldots\})$:

$$\overset{\emptyset}{\bullet} \longrightarrow \overset{\{b\}}{\bullet} \longrightarrow \overset{\{a,b\}}{\bullet} \longrightarrow \overset{\emptyset}{\bullet} \longrightarrow \overset{\{b\}}{\bullet}$$

We will note $\Sigma = \{a, b, \ldots\}$, and $A = \mathcal{P}(\Sigma)$ the alphabet.

- Useful e.g. in verification (LTL,...):
  independent signals can be true or false simultaneously.
- FO languages on alphabet A are the same (Preds$=\Sigma$ or A).

FO on words, the "unconstrained" way

For now, a word is a structure $(X, \leq, a, b, \dots)$ where

- $\leq$ is a total order
- ~~$a, b, \dots$ form a partition of X.~~

$a, b, \dots$ now independent.
$\rightarrow$ Words on alphabet $\mathcal{P}(\{a, b, \dots\})$:

$$\overset{\emptyset}{\bullet} \longrightarrow \overset{\{b\}}{\bullet} \longrightarrow \overset{\{a,b\}}{\bullet} \longrightarrow \overset{\emptyset}{\bullet} \longrightarrow \overset{\{b\}}{\bullet}$$

We will note $\Sigma = \{a, b, \dots\}$, and $A = \mathcal{P}(\Sigma)$ the alphabet.

- Useful e.g. in verification (LTL,...):
  independent signals can be true or false simultaneously.
- FO languages on alphabet A are the same (Preds$=\Sigma$ or A).
- We no longer have $\neg a(x) \equiv \bigvee_{\beta \neq a} \beta(x)$.

FO on words, the "unconstrained" way

For now, a word is a structure $(X, \leq, a, b, \dots)$ where

- $\leq$ is a total order
- ~~$a, b, \dots$ form a partition of X.~~

$a, b, \dots$ now independent.
$\rightarrow$ Words on alphabet $\mathcal{P}(\{a, b, \dots\})$:

$$\begin{array}{ccccccccc} \emptyset & & \{b\} & & \{a,b\} & & \emptyset & & \{b\} \\ \bullet & \longrightarrow & \bullet & \longrightarrow & \bullet & \longrightarrow & \bullet & \longrightarrow & \bullet \end{array}$$

We will note $\Sigma = \{a, b, \dots\}$, and $A = \mathcal{P}(\Sigma)$ the alphabet.

- Useful e.g. in verification (LTL,...):
  independent signals can be true or false simultaneously.
- FO languages on alphabet A are the same (Preds$=\Sigma$ or A).
- We no longer have $\neg a(x) \equiv \bigvee_{\beta \neq a} \beta(x)$.
  $\rightarrow$ Negation necessary for full FO.

The FO^+ logic: positive formulas

FO^+ Logic: a ranges over Σ, no $\neg$

$$\varphi, \psi := a(x) \mid x \leq y \mid x < y \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \exists x.\varphi \mid \forall x.\varphi$$

The FO^+ logic: positive formulas

FO^+ Logic: a ranges over Σ, no $\neg$

$$\varphi, \psi := a(x) \mid x \leq y \mid x < y \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \exists x.\varphi \mid \forall x.\varphi$$

Example: On $\Sigma = \{a, b\}$:

$$\exists x, y.(x \leq y) \wedge a(x) \wedge b(y) \quad \rightsquigarrow \quad (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*)$$

The FO^+ logic: positive formulas

FO^+ Logic: a ranges over Σ, no $\neg$

$$\varphi, \psi := a(x) \mid x \leq y \mid x < y \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \exists x.\varphi \mid \forall x.\varphi$$

Example: On $\Sigma = \{a, b\}$:

$$\exists x, y.(x \leq y) \wedge a(x) \wedge b(y) \quad \rightsquigarrow \quad (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*)$$

Remark: $\emptyset^*$ undefinable in FO^+ (cannot say "$\neg a$").

The FO^+ logic: positive formulas

FO^+ Logic: a ranges over Σ, no $\neg$

$$\varphi, \psi := a(x) \mid x \leq y \mid x < y \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \exists x.\varphi \mid \forall x.\varphi$$

Example: On $\Sigma = \{a, b\}$:

$$\exists x, y.(x \leq y) \wedge a(x) \wedge b(y) \quad \rightsquigarrow \quad (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*)$$

Remark: $\emptyset^*$ undefinable in FO^+ (cannot say "$\neg a$").

More generally: FO^+ can only define monotone languages:

$$u\alpha v \in L \text{ and } \alpha \subseteq \beta \Rightarrow u\beta v \in L$$

The FO^+ logic: positive formulas

FO^+ Logic: a ranges over Σ, no $\neg$

$$\varphi, \psi := a(x) \mid x \leq y \mid x < y \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \exists x.\varphi \mid \forall x.\varphi$$

Example: On $\Sigma = \{a, b\}$:

$$\exists x, y.(x \leq y) \wedge a(x) \wedge b(y) \quad \rightsquigarrow \quad (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*)$$

Remark: $\emptyset^*$ undefinable in FO^+ (cannot say "$\neg a$").

More generally: FO^+ can only define monotone languages:

$$u\alpha v \in L \text{ and } \alpha \subseteq \beta \Rightarrow u\beta v \in L$$

Motivation: abstraction of many logics not closed under $\neg$.

The FO^+ logic: positive formulas

FO^+ Logic: a ranges over Σ, no $\neg$

$$\varphi, \psi := a(x) \mid x \leq y \mid x < y \mid \varphi \vee \psi \mid \varphi \wedge \psi \mid \exists x.\varphi \mid \forall x.\varphi$$

Example: On $\Sigma = \{a, b\}$:

$$\exists x, y.(x \leq y) \wedge a(x) \wedge b(y) \quad \rightsquigarrow \quad (A^*\{a\}A^*\{b\}A^*) \cup (A^*\{a, b\}A^*)$$

Remark: $\emptyset^*$ undefinable in FO^+ (cannot say "$\neg a$").

More generally: FO^+ can only define monotone languages:

$$u\alpha v \in L \text{ and } \alpha \subseteq \beta \Rightarrow u\beta v \in L$$

Motivation: abstraction of many logics not closed under $\neg$.

Question [Colcombet]: FO & monotone $\overset{?}{\Rightarrow}$ FO^+

A counter-example language

Our first result

There is L monotone, FO-definable but not FO^+-definable.

A counter-example language

Our first result

There is L monotone, FO-definable but not FO^+-definable.

Alphabet $A = \{\emptyset, a, b, c, \binom{a}{b}, \binom{b}{c}, \binom{c}{a}, \begin{pmatrix} a \\ b \\ c \end{pmatrix}\}$. Let $a^{\uparrow} = \{a, \binom{a}{b}, \binom{c}{a}\}$.

A counter-example language

Our first result

There is L monotone, FO-definable but not FO^+-definable.

Alphabet $A = \{\emptyset, a, b, c, \binom{a}{b}, \binom{b}{c}, \binom{c}{a}, \begin{pmatrix} a \\ b \\ c \end{pmatrix}\}$. Let $a^\uparrow = \{a, \binom{a}{b}, \binom{c}{a}\}$.

Language $L = (a^\uparrow b^\uparrow c^\uparrow)^* \bigcup A^* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*$.

A counter-example language

Our first result

There is L monotone, FO-definable but not FO^+-definable.

Alphabet $A = \{\emptyset, a, b, c, \binom{a}{b}, \binom{b}{c}, \binom{c}{a}, \begin{pmatrix} a \\ b \\ c \end{pmatrix}\}$. Let $a^\uparrow = \{a, \binom{a}{b}, \binom{c}{a}\}$.

Language $\quad L = (a^\uparrow b^\uparrow c^\uparrow)^* \quad \bigcup \quad A^* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*.$ Monotone

A counter-example language

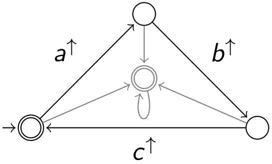
Our first result

There is L monotone, FO-definable but not FO$^+$-definable.

Alphabet $A = \{\emptyset, a, b, c, \binom{a}{b}, \binom{b}{c}, \binom{c}{a}, \begin{pmatrix} a \\ b \\ c \end{pmatrix}\}$. Let $a^\uparrow = \{a, \binom{a}{b}, \binom{c}{a}\}$.

Language $L = (a^\uparrow b^\uparrow c^\uparrow)^* \ \bigcup \ A^* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*$. Monotone

Lemma: L is FO-definable.

Proof: 

is counter-free. (no cycle labelled $v^{\geq 2}$)

A counter-example language

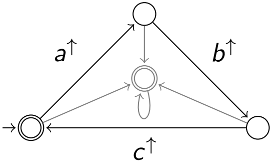
Our first result

There is L monotone, FO-definable but not FO^+-definable.

Alphabet $A = \{\emptyset, a, b, c, \binom{a}{b}, \binom{b}{c}, \binom{c}{a}, \begin{pmatrix} a \\ b \\ c \end{pmatrix}\}$. Let $a^\uparrow = \{a, \binom{a}{b}, \binom{c}{a}\}$.

Language $L = (a^\uparrow b^\uparrow c^\uparrow)^* \;\bigcup\; A^* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*$. Monotone

Lemma: L is FO-definable.

Proof: 

is counter-free. (no cycle labelled $v^{\geq 2}$)

To prove L is not FO^+-definable: Ehrenfeucht-Fraïssé games.

Ehrenfeucht-Fraïssé games for FO

Definition (EF games)

Played on two words u, v. At each round i:

- Spoiler places token i in u or v.
- Duplicator must answer token i in the other word such that
  - same letter for token i,
  - same order between tokens.

Ehrenfeucht-Fraïssé games for FO

Definition (EF games)

Played on two words u, v. At each round i:

- Spoiler places token i in u or v.
- Duplicator must answer token i in the other word such that
  - same letter for token i,
  - same order between tokens.

We note $u \equiv_n v$ if Duplicator can survive n rounds on u, v.

Ehrenfeucht-Fraïssé games for FO

Definition (EF games)

Played on two words u, v. At each round i:

- Spoiler places token i in u or v.
- Duplicator must answer token i in the other word such that
  - same letter for token i,
  - same order between tokens.

We note $u \equiv_n v$ if Duplicator can survive n rounds on u, v.

Theorem (Ehrenfeucht,Fraïssé, 1950-1961)

L not FO-definable $\Leftrightarrow$ For all n, there are $u \in L$, $v \notin L$ s.t. $u \equiv_n v$.

Ehrenfeucht-Fraïssé games for FO

Definition (EF games)

Played on two words u, v. At each round i:

- Spoiler places token i in u or v.
- Duplicator must answer token i in the other word such that
  - same letter for token i,
  - same order between tokens.

We note $u \equiv_n v$ if Duplicator can survive n rounds on u, v.

Theorem (Ehrenfeucht,Fraïssé, 1950-1961)

L not FO-definable $\Leftrightarrow$ For all n, there are $u \in L$, $v \notin L$ s.t. $u \equiv_n v$.

Example

Proving $(aa)^*$ is not FO-definable:

$$\begin{array}{lll} u = a^{2k} & \in (aa)^* : & a\ a\ a\ a\ a\ a\ a\ a\ a\ a \\ v = a^{2k-1} & \notin (aa)^* : & a\ a\ a\ a\ a\ a\ a\ a\ a \end{array}$$

Proving FO^+-undefinability

Definition (EF^+ games)

Previous rule: a in $u \Leftrightarrow a$ in v.

We write $u \preceq_n v$ if Duplicator can survive n rounds.

Proving FO^+-undefinability

Definition (EF^+ games)

New rule: a in $u \Rightarrow a$ in v.

We write $u \preceq_n v$ if Duplicator can survive n rounds.

Proving FO^+-undefinability

Definition (EF^+ games)

New rule: a in $u \Rightarrow a$ in v.

We write $u \preceq_n v$ if Duplicator can survive n rounds.

Theorem (Correctness of EF^+ games)

L not FO^+-definable $\Leftrightarrow \forall n$, there are $u \in L$, $v \notin L$ s.t. $u \preceq_n v$.
[Stolboushkin 1995+this work]

Proving FO^+-undefinability

Definition (EF^+ games)

New rule: a in $u \Rightarrow a$ in v.

We write $u \preceq_n v$ if Duplicator can survive n rounds.

Theorem (Correctness of EF^+ games)

L not FO^+-definable $\Leftrightarrow \forall n$, there are $u \in L$, $v \notin L$ s.t. $u \preceq_n v$.
[Stolboushkin 1995+this work]

Application: Proving L is not FO^+-definable

$$\begin{array}{llllllllll} u \in L: & a & b & c & a & b & c & a & b & c \\ v \notin L: & \binom{a}{b} & \binom{b}{c} & \binom{c}{a} & \binom{a}{b} & \binom{b}{c} & \binom{c}{a} & \binom{a}{b} & \binom{b}{c} \end{array}$$

From finite words to finite structures.

Goal: Lift L to finite structures.

For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

From finite words to finite structures.

Goal: Lift L to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

Several monotone predicates

Axiomatize in FO that $\leq$ is a total order.

From finite words to finite structures.

Goal: Lift L to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

Several monotone predicates

Axiomatize in FO that $\leq$ is a total order.
a, b, c are monotone but not $\leq$.

From finite words to finite structures.

Goal: Lift L to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

Several monotone predicates

Axiomatize in FO that $\leq$ is a total order.
a, b, c are monotone but not $\leq$.

One monotone predicate

Alphabet encoded by one binary predicate A.

$$a(x) \equiv A(0, x) \qquad b(x) \equiv A(1, x) \qquad c(x) \equiv A(2, x)$$

From finite words to finite structures.

Goal: Lift L to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

Several monotone predicates

Axiomatize in FO that $\leq$ is a total order.
a, b, c are monotone but not $\leq$.

One monotone predicate

Alphabet encoded by one binary predicate A.

$$a(x) \equiv A(0, x) \qquad b(x) \equiv A(1, x) \qquad c(x) \equiv A(2, x)$$

A is monotone but not $\leq$.

From finite words to finite structures.

Goal: Lift L to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

Several monotone predicates

Axiomatize in FO that $\leq$ is a total order.
a, b, c are monotone but not $\leq$.

One monotone predicate

Alphabet encoded by one binary predicate A.

$$a(x) \equiv A(0, x) \qquad b(x) \equiv A(1, x) \qquad c(x) \equiv A(2, x)$$

A is monotone but not $\leq$.

All monotone predicates = closure under surjective morphisms

Problem: We cannot say that $\leq$ is total in a monotone way.

From finite words to finite structures.

Goal: Lift L to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

Several monotone predicates

Axiomatize in FO that $\leq$ is a total order.
a, b, c are monotone but not $\leq$.

One monotone predicate

Alphabet encoded by one binary predicate A.

$$a(x) \equiv A(0, x) \qquad b(x) \equiv A(1, x) \qquad c(x) \equiv A(2, x)$$

A is monotone but not $\leq$.

All monotone predicates = closure under surjective morphisms

Problem: We cannot say that $\leq$ is total in a monotone way.
Solution: Introduce a predicate $\not\leq$.

From finite words to finite structures.

Goal: Lift L to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

Several monotone predicates

Axiomatize in FO that $\leq$ is a total order.
a, b, c are monotone but not $\leq$.

One monotone predicate

Alphabet encoded by one binary predicate A.

$$a(x) \equiv A(0, x) \qquad b(x) \equiv A(1, x) \qquad c(x) \equiv A(2, x)$$

A is monotone but not $\leq$.

All monotone predicates = closure under surjective morphisms

Problem: We cannot say that $\leq$ is total in a monotone way.
Solution: Introduce a predicate $\not\leq$.

- Require $\forall x, y.(x \leq y) \vee (x \not\leq y)$
- If $\exists x, y.(x \leq y) \wedge (x \not\leq y) \rightarrow$ accept
- Axiomatize that $\leq$ is total assuming $\not\leq$ is its complement.

From finite words to finite structures.

Goal: Lift L to finite structures.
For now: signature $(\leq, a, b, c)$ assuming $\leq$ is a total order.

Several monotone predicates

Axiomatize in FO that $\leq$ is a total order.
a, b, c are monotone but not $\leq$.

One monotone predicate

Alphabet encoded by one binary predicate A.

$$a(x) \equiv A(0, x) \qquad b(x) \equiv A(1, x) \qquad c(x) \equiv A(2, x)$$

A is monotone but not $\leq$.

All monotone predicates = closure under surjective morphisms

Problem: We cannot say that $\leq$ is total in a monotone way.
Solution: Introduce a predicate $\not\leq$.

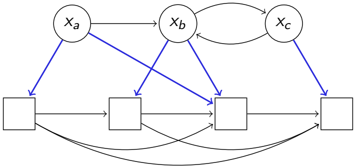
- Require $\forall x, y.(x \leq y) \vee (x \not\leq y)$
- If $\exists x, y.(x \leq y) \wedge (x \not\leq y) \rightarrow$ accept
- Axiomatize that $\leq$ is total assuming $\not\leq$ is its complement.

$a, b, c, \leq, \not\leq$ are monotone.

From finite words to finite graphs

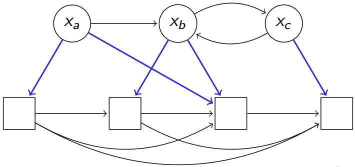
From finite words to finite graphs

Encode words into (directed) graphs, here $ab\binom{a}{b}c$:



From finite words to finite graphs

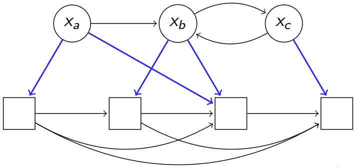
Encode words into (directed) graphs, here $ab\binom{a}{b}c$:



$\to$ formula ψ_L for graphs encoding words of $L = (a^{\uparrow} b^{\uparrow} c^{\uparrow})^* \cup (A^* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*)$.

From finite words to finite graphs

Encode words into (directed) graphs, here $ab\binom{a}{b}c$:



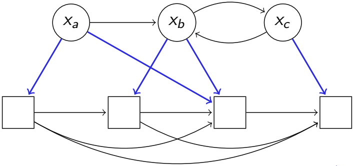
$\rightarrow$ formula ψ_L for graphs encoding words of $L = (a^{\uparrow} b^{\uparrow} c^{\uparrow})^* \cup (A^* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*)$.

Rule out other graphs, in a monotone way:

- ψ^- is a conjunction of edge requirements:

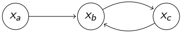
From finite words to finite graphs

Encode words into (directed) graphs, here $ab\binom{a}{b}c$:



$\rightarrow$ formula ψ_L for graphs encoding words of $L = (a^{\uparrow} b^{\uparrow} c^{\uparrow})^* \cup (A^* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*)$.

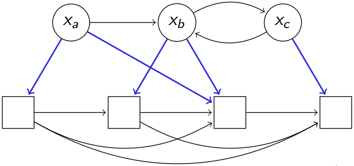
Rule out other graphs, in a monotone way:

- ψ^- is a conjunction of edge requirements:
  - 

,

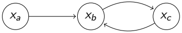
From finite words to finite graphs

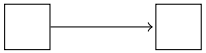
Encode words into (directed) graphs, here $ab\binom{a}{b}c$:



$\rightarrow$ formula ψ_L for graphs encoding words of $L = (a^\uparrow b^\uparrow c^\uparrow)^* \cup (A^* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*)$.

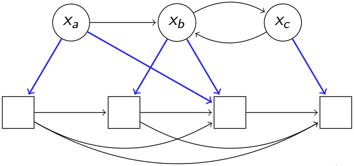
Rule out other graphs, in a monotone way:

- ψ^- is a conjunction of edge requirements:
  - 

,
  - 
,...

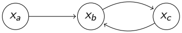
From finite words to finite graphs

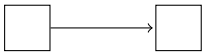
Encode words into (directed) graphs, here $ab\binom{a}{b}c$:



$\rightarrow$ formula ψ_L for graphs encoding words of $L = (a^{\uparrow} b^{\uparrow} c^{\uparrow})^* \cup (A^* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*)$.

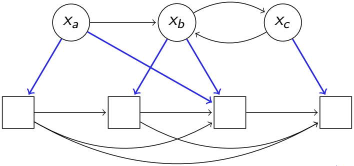
Rule out other graphs, in a monotone way:

- ψ^- is a conjunction of edge requirements:
    - 

,
    - 
,...
- ψ^+ is a disjunction of excess edges:

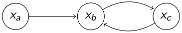
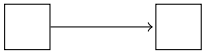
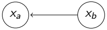
From finite words to finite graphs

Encode words into (directed) graphs, here $ab\binom{a}{b}c$:



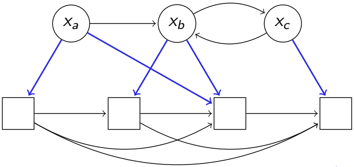
$\to$ formula ψ_L for graphs encoding words of $L = (a^{\uparrow} b^{\uparrow} c^{\uparrow})^* \cup (A^* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*)$.

Rule out other graphs, in a monotone way:

- ψ^- is a conjunction of edge requirements:
  - 
,
  - ,...
- ψ^+ is a disjunction of excess edges:
  - 
,

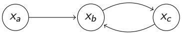
From finite words to finite graphs

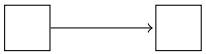
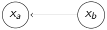
Encode words into (directed) graphs, here $ab\binom{a}{b}c$:

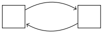


$\rightarrow$ formula ψ_L for graphs encoding words of $L = (a^\uparrow b^\uparrow c^\uparrow)^* \cup (A^* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*)$.

Rule out other graphs, in a monotone way:

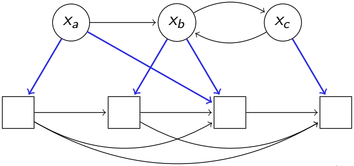
- ψ^- is a conjunction of edge requirements:
  - 

    ,
  - 
    ,...
- ψ^+ is a disjunction of excess edges:
  - 

    ,
  - 
    ,...

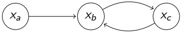
From finite words to finite graphs

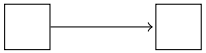
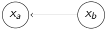
Encode words into (directed) graphs, here $ab\binom{a}{b}c$:

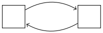


$\rightarrow$ formula ψ_L for graphs encoding words of $L = (a^\uparrow b^\uparrow c^\uparrow)^* \cup (A^* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*)$.

Rule out other graphs, in a monotone way:

- ψ^- is a conjunction of edge requirements:
  - 

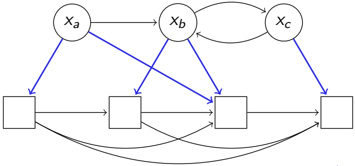
    ,
  - 
    ,...
- ψ^+ is a disjunction of excess edges:
  - 

    ,
  - 
    ,...

Final Formula: $\exists x_a, x_b, x_c.(\psi^- \wedge (\psi_L \vee \psi^+))$

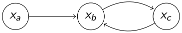
From finite words to finite graphs

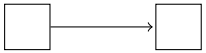
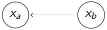
Encode words into (directed) graphs, here $ab\binom{a}{b}c$:

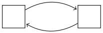


$\to$ formula ψ_L for graphs encoding words of $L = (a^{\uparrow} b^{\uparrow} c^{\uparrow})^* \cup (A^* \begin{pmatrix} a \\ b \\ c \end{pmatrix} A^*)$.

Rule out other graphs, in a monotone way:

- ψ^- is a conjunction of edge requirements:
  - 

,
  - ,...
- ψ^+ is a disjunction of excess edges:
  - 

,
  - ,...

Final Formula: $\exists x_a, x_b, x_c.(\psi^- \wedge (\psi_L \vee \psi^+))$

Left as exercise: Same with undirected graphs.

Back to regular languages

Theorem
Given L regular on an ordered alphabet, it is decidable whether

- *L is monotone (e.g. automata inclusion)*
- *L is FO-definable [Schützenberger, McNaughton, Papert]*

Can we decide whether L is FO^{+}-definable ?

Back to regular languages

Theorem

Given L regular on an ordered alphabet, it is decidable whether

- *L is monotone (e.g. automata inclusion)*
- *L is FO-definable [Schützenberger, McNaughton, Papert]*

Can we decide whether L is FO^+-definable ?

Theorem

FO^+-definability is undecidable for regular languages.

Back to regular languages

Theorem

Given L regular on an ordered alphabet, it is decidable whether

- *L is monotone (e.g. automata inclusion)*
- *L is FO-definable [Schützenberger, McNaughton, Papert]*

Can we decide whether L is FO^+-definable ?

Theorem

FO^+-definability is undecidable for regular languages.

Reduction from *Turing Machine Mortality*:

A deterministic TM M is *mortal* if there a uniform bound n on the runs of M from **any** configuration.

Undecidable [Hooper 1966].

Undecidability proof sketch

Given a TM M, we build a regular language L such that

$$M \text{ mortal } \Leftrightarrow L \text{ is } \mathrm{FO}^+\text{-definable.}$$

Undecidability proof sketch

Given a TM M, we build a regular language L such that

$$M \text{ mortal } \Leftrightarrow L \text{ is } \mathrm{FO}^+\text{-definable.}$$

Building L:
Inspired from $(a^\uparrow b^\uparrow c^\uparrow)^*$, but:

- $a, b, c \rightsquigarrow$ Words from languages C_1, C_2, C_3 encoding configs of M.

- All transitions of M follow the cycle:
$$\begin{tikzcd} & C_2 \arrow[dr] & \\ C_1 \arrow[ur] & & C_3 \arrow[ll] \end{tikzcd}$$

- $\binom{a}{b}, \binom{b}{c}, \binom{c}{a} \rightsquigarrow \binom{u_1}{u_2}$, exists iff $u_1 \xrightarrow{M} u_2$.

Undecidability proof sketch

Given a TM M, we build a regular language L such that

$$M \text{ mortal } \Leftrightarrow L \text{ is } \mathrm{FO}^+\text{-definable.}$$

Building L:
Inspired from $(a^\uparrow b^\uparrow c^\uparrow)^*$, but:

- $a, b, c \rightsquigarrow$ Words from languages C_1, C_2, C_3 encoding configs of M.
- All transitions of M follow the cycle:
$$\begin{tikzcd} & C_2 \arrow[dr] & \\ C_1 \arrow[ur] & & C_3 \arrow[ll] \end{tikzcd}$$
- $\binom{a}{b}, \binom{b}{c}, \binom{c}{a} \rightsquigarrow \binom{u_1}{u_2}$, exists iff $u_1 \xrightarrow{M} u_2$.

We choose

$$L := (C_1^\uparrow \cdot C_2^\uparrow \cdot C_3^\uparrow)^*$$

Undecidability proof sketch

Given a TM M, we build a regular language L such that

$$M \text{ mortal } \Leftrightarrow L \text{ is } \mathrm{FO}^+\text{-definable.}$$

Building L:
Inspired from $(a^\uparrow b^\uparrow c^\uparrow)^*$, but:

- $a, b, c \rightsquigarrow$ Words from languages C_1, C_2, C_3 encoding configs of M.
- All transitions of M follow the cycle:

$$\begin{tikzcd} & C_2 \arrow[dr] & \\ C_1 \arrow[ur] & & C_3 \arrow[ll] \end{tikzcd}$$

- $\binom{a}{b}, \binom{b}{c}, \binom{c}{a} \rightsquigarrow \binom{u_1}{u_2}$, exists iff $u_1 \xrightarrow{M} u_2$.

We choose

$$L := (C_1^\uparrow \cdot C_2^\uparrow \cdot C_3^\uparrow)^*$$

$u \in L \not\Rightarrow u$ encodes a run of M.

The reduction

If M not mortal:
Let $u_1, u_2, \ldots, u_n$ a long run of M, and play Duplicator in :

$$\begin{array}{llllllll} u \in L: & u_1 & u_2 & u_3 & \ldots & u_{n-1} & u_n \\ v \notin L: & \binom{u_1}{u_2} & \binom{u_2}{u_3} & \binom{u_3}{u_4} & \ldots & \binom{u_{n-1}}{u_n} & \end{array}$$

$\rightarrow$ L is not FO^+-definable.

The reduction

If M not mortal:
Let $u_1, u_2, \ldots, u_n$ a long run of M, and play Duplicator in :

$$\begin{array}{llllllll} u \in L: & u_1 & u_2 & u_3 & \ldots & u_{n-1} & u_n \\ v \notin L: & \binom{u_1}{u_2} & \binom{u_2}{u_3} & \binom{u_3}{u_4} & \ldots & \binom{u_{n-1}}{u_n} & \end{array}$$

$\rightarrow L$ is not FO^+-definable.

If M mortal with bound n:
Abstract u_i by the length of the run of M starting in it (at most n).

The reduction

If M not mortal:
Let $u_1, u_2, \ldots, u_n$ a long run of M, and play Duplicator in :

$$\begin{array}{llllllll} u \in L: & u_1 & u_2 & u_3 & \ldots & u_{n-1} & u_n \\ v \notin L: & \binom{u_1}{u_2} & \binom{u_2}{u_3} & \binom{u_3}{u_4} & \ldots & \binom{u_{n-1}}{u_n} & \end{array}$$

$\rightarrow L$ is not FO^+-definable.

If M mortal with bound n:
Abstract u_i by the length of the run of M starting in it (at most n).

Play Spoiler in the abstracted game (here $n = 5$):

$$\begin{array}{lcccccccccc} u: & 2 & 3 & 2 & 4 & 3 & 5 & 4 & 3 & 4 & 4 \\ v: & \binom{2}{1} & \binom{3}{2} & \binom{2}{1} & \binom{4}{3} & \binom{3}{2} & \binom{5}{4} & \binom{4}{3} & \binom{5}{4} & \binom{5}{4} & \end{array}$$

The reduction

If M not mortal:
Let $u_1, u_2, \ldots, u_n$ a long run of M, and play Duplicator in :

$$\begin{array}{llllllll} u \in L: & u_1 & u_2 & u_3 & \ldots & u_{n-1} & u_n \\ v \notin L: & \binom{u_1}{u_2} & \binom{u_2}{u_3} & \binom{u_3}{u_4} & \ldots & \binom{u_{n-1}}{u_n} & \end{array}$$

$\rightarrow L$ is not FO^+-definable.

If M mortal with bound n:
Abstract u_i by the length of the run of M starting in it (at most n).

Play Spoiler in the abstracted game (here $n = 5$):

$$\begin{array}{lcccccccccc} u: & 2 & 3 & 2 & 4 & 3 & 5 & 4 & 3 & 4 & 4 \\ v: & \binom{2}{1} & \binom{3}{2} & \binom{2}{1} & \binom{4}{3} & \binom{3}{2} & \binom{5}{4} & \binom{4}{3} & \binom{5}{4} & \binom{5}{4} & \end{array}$$

Spoiler always wins in $2n$ rounds $\rightarrow L$ is FO^+-definable.

Ongoing work

With Thomas Colcombet:
Exploring the consequences of this in other frameworks:

- regular cost functions,
- logics on linear orders,
- ...

With Quentin Moreau:

- Links with LTL
- FO2 fragment
- ...

Slogan:
FO variants without negation will often display this behaviour.

Ongoing work

With Thomas Colcombet:
Exploring the consequences of this in other frameworks:

- regular cost functions,
- logics on linear orders,
- ...

With Quentin Moreau:

- Links with LTL
- FO2 fragment
- ...

Slogan:
FO variants without negation will often display this behaviour.

Thanks for your attention !